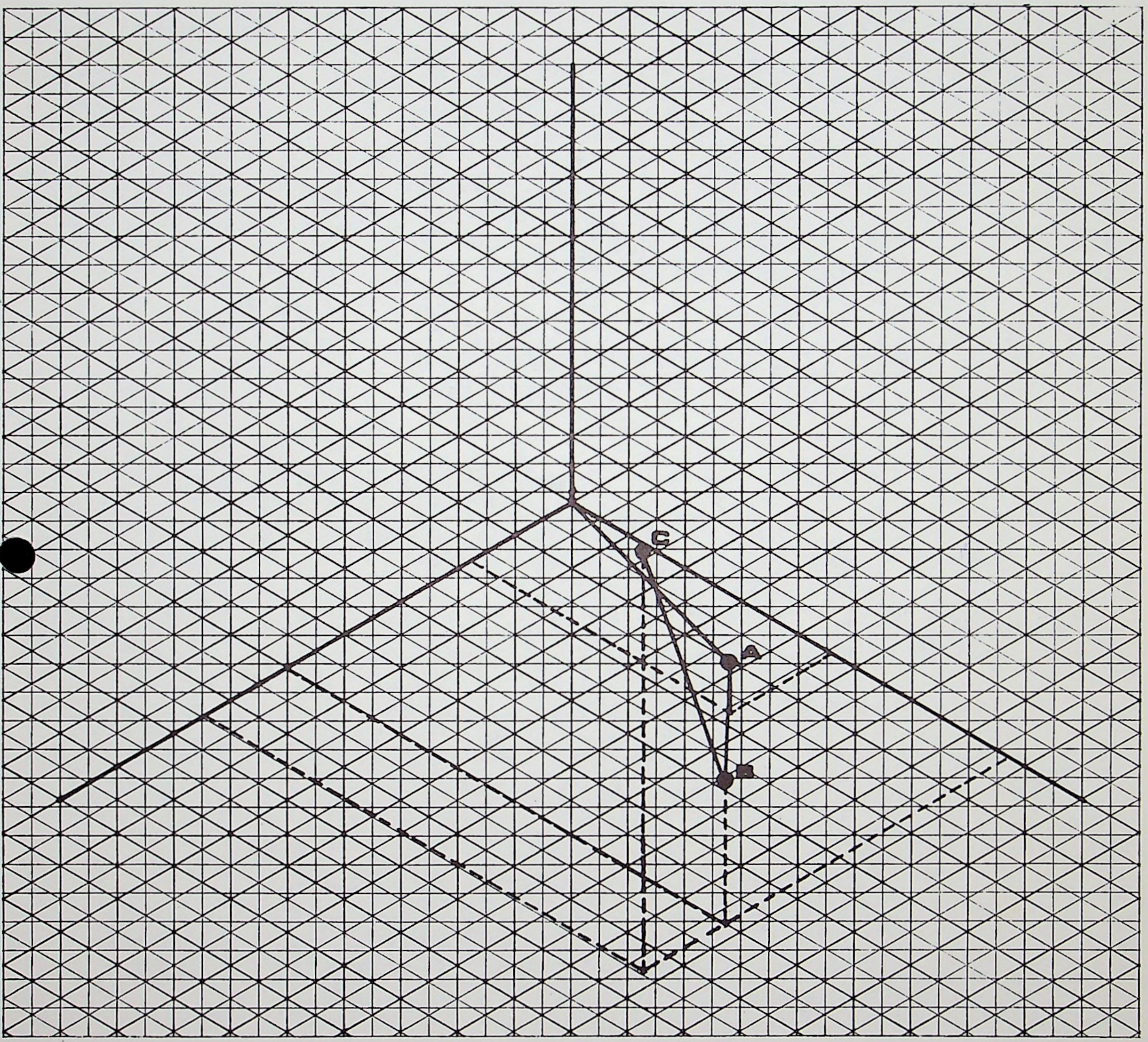
BOX 272
CALABASAS CA
91302

Popular Computing

VOL 2 NO 7
JULY 74



Cube Route

PROBLEM 52

Cube Route

Starting with issue No. 6 (September 1973), every white issue has featured a journey:

6: The Pi Dragon
8: The Road to e
10: The Web of Fibonacci
12: The Sine Excursion
14: The Change of Base

The trip this time is 3-dimensional, using the digits of the cube root of 16. Taking the digits 3 at a time:

251 984 209 978 974 632 953 ...

each triplet is taken as the direction numbers of a line segment of unit length. Starting at the origin, we advance to point A, whose coordinates are:

$$X = \frac{2}{\sqrt{2^2 + 5^2 + 1^2}}$$

$$Y = \frac{5}{\sqrt{30}}$$

$$Z = \frac{1}{\sqrt{30}}$$

that is, to the point (.3651483, .9128709, .1825741).

Point B in the trip is one unit from A, in the direction given by (9,8,4); that is, we add to the coordinates of A the quantities

$$\frac{9}{\sqrt{161}} \qquad \frac{8}{\sqrt{161}} \qquad \frac{4}{\sqrt{161}}$$

So that the coordinates of B are:

$$X = .3651483 + .709299 = 1.074447$$

$$Y = .9128709 + .630488 = 1.543359$$

$$Z = .1825741 + .315244 = .497818$$

POPULAR COMPUTING is published monthly at Box 272, Calabasas, California 91302. Subscription rate in the United States is $18 per year, or $15 if remittance accompanies the order. For Canada and Mexico, add $4 to the above rates. For all other countries, add $6 to the above rates. Back issues $2 each. Subscriptions may begin with any issue.

Similarly, point C is one unit from B in the direction given by (2,0,9), or

$$X = 1.291377$$

$$Y = 1.543359$$

$$Z = 1.474005$$

Where is the 200th point of the CUBE ROUTE? (The first 1001 digits of the cube root of 16 can be found in this month's N-Series.)

*

TRICKS, GAMES, AND PUZZLES WANTED...

(Suitable for pocket calculators)

...for inclusion in a forthcoming book on games for pocket calculators. No payment can be made, but original contributors will be credited in the book. Write to:

James T. Rogers
Box 262
Sugar Loaf, NY 10981

*

CORRECTION: The review of The Elements of Programming Style (PC13-8) incorrectly listed the authors:

Brian W. Kernighan
P. J. Plauger

*

ART OF COMPUTING 3

FJG

THE COMPUTER AS A PROBLEM SOLVING TOOL

In a strict sense, computers don't solve problems at all. People solve problems, and express the solution in an algorithm called a program. The computer, in executing such a program, lets the user play out the consequences of his decisions. In performing this service, the computer can quickly reveal an incorrect solution, or one so illogical that the program hangs up. But even a logically correct solution can be a poor solution, and the speed of the machine can often reveal this fact.

The computer's speed is the superficial characteristic that immediately differentiates it from lesser and earlier calculating devices. Speed is essential, in order to play out those consequences while the user is still interested in the problem being solved, but it is only the tip of the iceberg. The ability to sequence a predetermined set of instructions makes the machine automatic, and the ability to jump control permits the path through the instructions to be altered based on decisions that are made dynamically as the calculation proceeds. But all of this describes only a sequenced <u>calculator</u>. The essence of the computer is its ability to modify its own instructions. The lowest level of this capability is simple address modification, to facilitate references to different data items during each traverse of a loop. But the ability to have one instruction act on another instruction, treating the latter as data (which it then is) is a powerful tool. It allows for such things as programmed switches, short paths to the selection of one of many subroutines, efficient searching of tables, efficient handling of arrays, variable length fields and records, relocatable code, and hash coding schemes.

In other words, the computer is not <u>just</u> a faster and bigger calculator (though it is certainly that), but a new tool for problem solving. Which is not to say that it has superseded other older tools, but rather has amplified them and supplemented them. When all else fails, the computer offers the possibility of solving a problem by sheer brute force; that is, by exhaustion. Crowbar methods should be a last resort, but the computer at least makes such methods cheap, and hence feasible.

There is no magic method for solving any problem. Those who do it best are usually those who have done it the most; it is an art that develops with practice. But some guidelines can be stated.

1. Analyze and define the problem. Is it a computer problem at all? Can a method of solution be found or devised? Has a problem like it been solved before? What type of problem is it? Can a way be found to get started? That is, is there a way to produce some results, however inefficiently? Have the various alternatives to using the computer (graphical methods; punched card methods; analytic solutions; paper-and-pencil methods) been carefully considered?

2. Flowchart the proposed method of solution. Clean up the logic on the flowchart. Look around for shortcuts (not in the subsequent coding!--in the logic). Documentation of the solution begins here, as does the matter of testing the eventual program.

3. Give some thought to the problems of data representation and data entry at this time. The data that you may blithely assume will be in storage has to get there somehow.

4. Code and debug the proposed solution. There may be a small problem of choice of language at this point. You may be constrained by the limited range of language translators available at your installation, but it should not be a constraint that you limit yourself to one language. The end result of all coding languages is machine language; assemblers, compilers, and generators are only different ways of getting there. Your problem solution will ultimately be executed by a machine that is built to process binary numbers, and only integers at that. The problem being solved may not involve numbers at all, but the computer solution deals entirely in numbers.

At this stage of the solution to a new problem, one should avoid the temptation to show off skill in coding. Clever tricks and coy techniques in the early stages will generally cause trouble and delays. The straightforward, simple-minded approach gets there faster; at this stage, what is wanted are quick and dirty results. The clever coding tricks can always be added later to a program that is known to work.

5. Make some trial runs of limited scope. Clean up the trial solution. Look for more shortcuts in the logic of the solution at this time; the trial run may reveal patterns that suggest a better solution. Get some timing information from the trial run; it may be that a complete run with the present logic would consume an inordinate amount of computer time. Don't be afraid to abandon the first approach and start all over. There is always a great temptation to become enamored of the first approach and cling to it when it is leading no where. But the first approach should still be tested to be sure that it is doing its job correctly.

6. Don't worry at this stage about perfect solutions. Perhaps a partial solution will serve. Don't work for 1% improvements--look for speedup factors of 2 or more.

Feel free to write on printouts with a pen; neat column headings that are produced by the program do not contribute to the solution now. Worry a lot at this stage about documentation of the solution. Are your flowcharts clear to others? Can your code be followed by someone else? Are there still coy tricks left in your code, or is it nice and simple and self-explanatory? Follow your own flowcharts, and have your code do exactly what the flowchart says. If you alter the logic of the solution at this point, change the flowcharts to conform.

7. Whenever you're stuck, try doing a simpler problem first, and then generalize from that.

8. Or, when you're stuck, try doing <u>something</u>, rather than nothing. Extend the solution by hand for a few more cases (this will also be useful in testing the computer solution). Try a slightly longer machine run, to try to gain more insight into the trial solution.

9. It is inappropriate to try for elegance in the first solution. An elegant solution is satisfying and may be far more efficient, but a crude solution may get one started, and may wind up saving elapsed time.

Consider three solutions to the same (textbook) problem, proposed by Dr. Richard Hamming (and given in PC10-8). Generate and print, in order, a list of integers having only the factors 2, 3, and/or 5; that is, a list that begins 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, 18, 20, 24, 25, 27, 30, 32, 36,...

Flowchart A shows a straightforward solution. The logic of the flowchart proceeds directly from the statement of the problem. Generate consecutive integers. Make a copy of each generated integer (following the precept that if any process destroys a number that needs to be saved, make a copy and destroy the copy). Then, for the copy, take out all possible factors of 2, of 3, and of 5. If this process reduces the copy to 1, print the original number and go on to the next integer.

Flowchart B (by Don Wanless) is less straightforward, and perhaps somewhat more elegant.

The third solution comes from Dr. Hamming himself, and is better described in words. Print and store the numbers 2, 3, 4, 5, and 6 as a starting list. Establish in storage the following values:

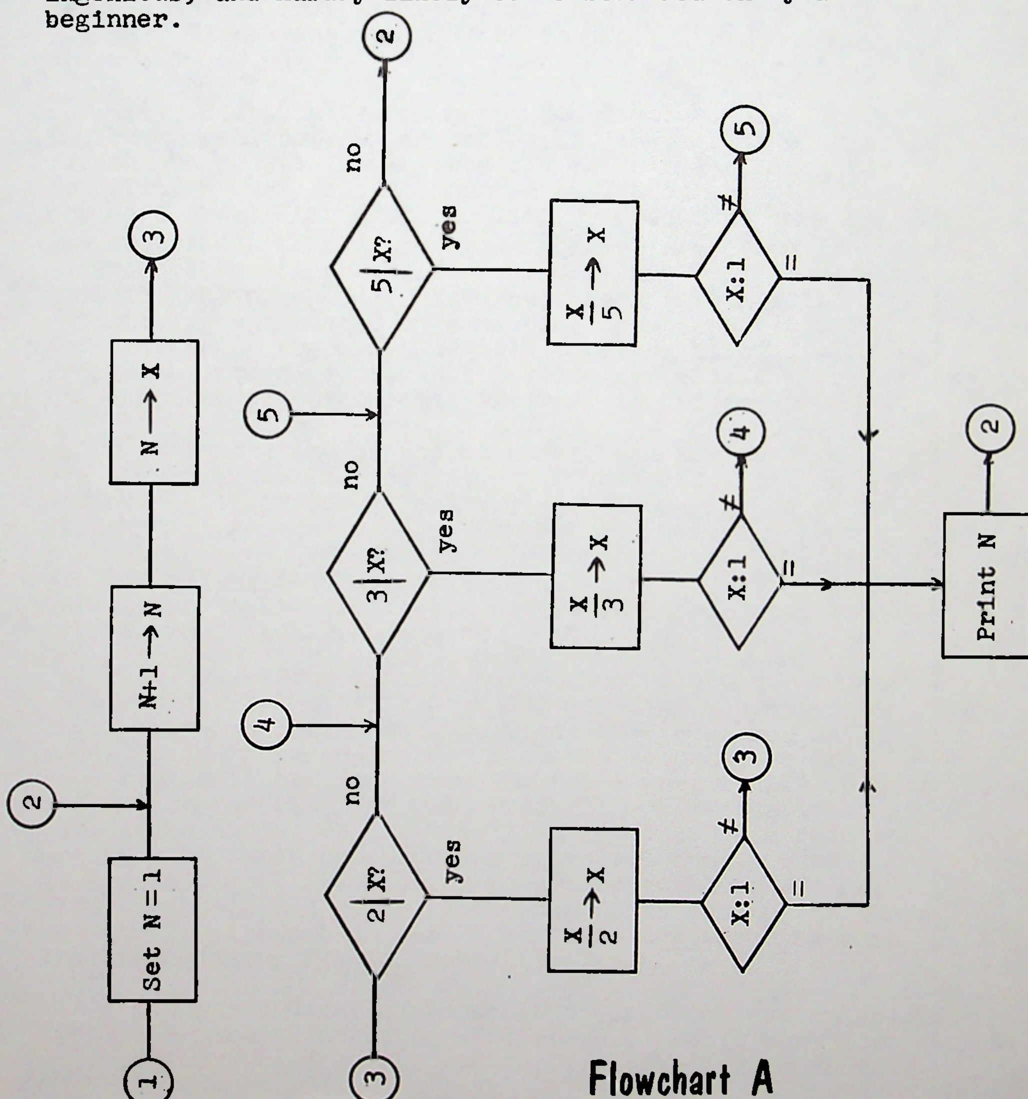
X	Y	Z	5X	3Y	2Z
2	3	4	10	9	8

Select the smallest of 5X, 3Y, 2Z and print it and add it to the list. Whichever variable (or variables) furnished that new result, advance it by the next item on the list. Recalculate 5X, 3Y, and 2Z and repeat. The six items

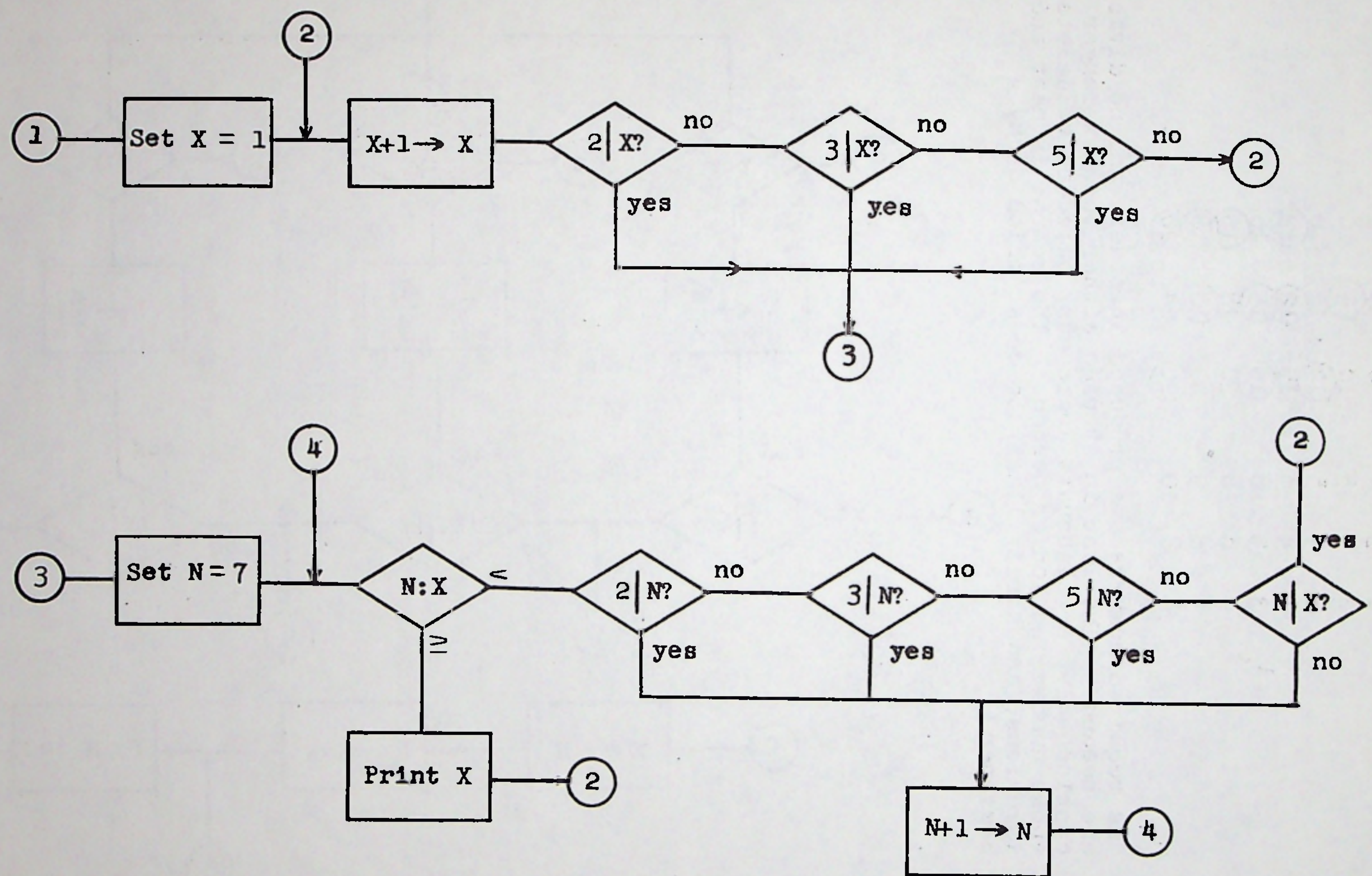
in storage change as follows:

X	Y	Z	5X	3Y	2Z
2	3	4	10	9	8
2	3	5	10	9	10
2	4	5	10	12	10
3	4	6	15	12	12
3	5	8	15	15	16
4	6	8	20	18	16
4	6	9	20	18	18
4	8	10	20	24	20

The required numbers are produced in order, and no divisions are needed. The list in storage either becomes longer continuously, or elaborate logic must be added to delete those numbers no longer needed. The scheme is neat and ingenious, and hardly likely to be stumbled on by a beginner.



Flowchart A


1
Set X = 1
2
X+1→ X
2|X?
no
yes
3|X?
no
yes
5|X?
no
yes
2
3
3
Set N=7
4
N:X
<
≥
Print X
2
2|N?
no
yes
3|N?
no
yes
5|N?
no
yes
N|X?
yes
no
2
N+1 → N
4

Flowchart B

10. A computer solution to a problem adds nothing that could not be done, in theory, by hand. Indeed, one method of testing a program is to reproduce some of its results by hand methods. This is not to say that the two approaches need be the same. We can use straightforward (crowbar) techniques with the computer that would be tedious or impossible by hand, simply in the interests of optimizing the overall solution; that is, to reduce one or more of programmer time, debugging time, compile time, elapsed time, and CPU time.

11. A computer solution should offer distinct advantages over other methods besides the obvious ones of speed and reliability. The computer solution should reduce costs and guarantee the accuracy of the results. It should also be human engineered. There are few programs that do not interface with humans, either at the input (data) end or the output (results) end, and a program that does not facilitate this interaction may be useless. Someone is going to have to read your printouts, so label them. No law says this cannot be done with a felt pen.

12. Above all, devise a scheme to test the finished program (see the article on Program Testing in PC-11).

Walk through the flowchart, playing like a computer, using selected values that will exercise the critical parts of the logic. Then arrange a procedure that will permit the computer to exercise the complete program for some limited range that can be validated against calculations made independently. Production runs should not be authorized until the programmer has reached some level of confidence that the program performs as advertised.

Speaking of Languages ROBERT TEAGUE

Since this column first appeared in September (PC6), there have been several requests for another language quiz. The following ten questions test your knowledge of Fortran, COBOL, Algol, and PL/1. Send your answers to "Speaking of Languages..," POPULAR COMPUTING, Box 272, Calabasas, California 91302. The earliest postmark will help determine the winner, to be declared in a future issue. (For the last quiz, there were no perfect papers.)

1. There are two types of data aggregates found in programming languages--name them. For each of Fortran, COBOL, Algol, and PL/1 indicate which are permitted.

2. Is the following a legal Fortran statement? Why?

```
DØ 10 I = 1.3
```

3. There are four major data types for PL/1 variables: Automatic, Static, Controlled, and Based.

a) Which is the default?
b) Which are specifically allocated by executable statements?
c) What happens when storage is reallocated prior to being released?

4. What is a keyword? Reserved word? Noise word? Which are found in Fortran, COBOL, Algol, and PL/1?

5. Recode the following PL/1 statement into Fortran, COBOL, and Algol:

```
DØ J = 1 TØ 9 WHILE (A>B);
```

6. Recode the following Algol loop into Fortran:

```
for L := 8, L+2 while L ≠ 150 do
A[L] := L / A[L];
```

7. How many ways can the variable C be qualified in PL/1?

```
DCL 1 A;
      2 B;
        3 C FIXED BINARY (15,0);
```

8. How many ways can the variable C be qualified in COBOL?

```
01 A.
   02 AA.
      03 B.
         04 C PICTURE 99.
```

9. How would the following storage allocation in COBOL be coded in Fortran and PL/1?

```
01 A.
   02 FILLER PICTURE 99.
01 B REDEFINES A.
   02 FILLER PICTURE 99.
```

10. Write a non-executable statement in Fortran to accomplish the following PL/1 loop.

```
DØ I = 1 TØ 100;
A(I) = I>50;
END;
```

16 Little Ones

PROBLEM 53 (A-P)

(A) Write a Fortran DØ loop that will operate on elements 1 to 100 of an array in either ascending or descending order of the subscripts depending on whether variable M is odd or even.

(B) It is required to calculate 3^x where x is large. If x = 1000, for example, we could perform 999 multiplications, or, using x = 1111101000 in binary, we could obtain 3^x in 5 multiplications, given a table of powers of powers of 3. (We want the exact value of 3^x and we can assume that any multiplication desired can be carried out.) The question is, which way is more efficient in terms of execution time? (The answer depends, of course, on the size of x.)

(C) Dr. Richard Hamming, in his book Computers and Society (McGraw-Hill, 1972) poses the following problem:

> "Suppose we start with a circle of radius 1, and we draw around it an equilateral triangle (circumscribed triangle), and around the triangle we draw another circle. Around this circle we draw a square, and around the square we draw another circle. Then we draw a regular five-sided polygon and another circle, then a regular six-sided polygon and a circle, and so on, continuing indefinitely. How will the radii of the circles behave? Will they approach infinity (get arbitrarily large), or will they remain bounded by some finite size?"

Using the notation R_3 for the radius of the circle that circumscribes the equilateral triangle, it is easy to show that

$$R_N = \frac{R_{N-1}}{\cos\,(180/N)}$$

and that R_N tends toward the value 8.6996 as N increases. The problem is to produce a more precise value of the limit.

(Problems D through I come from Professor Richard Andree)

D. Form a fraction, .14765..., in which each decimal place is the units' position of N^N. Is the result rational or irrational?

E. For any positive integer B that is not a power of 10 (i.e., B $\neq$ 1, 10, 100,...); given any positive integer K, there exists a power of B such that it begins with the digits of K. For the values:

$$B = 2, 3, 4, \ldots, 9$$
$$K = 1, 2, 3, \ldots, 100$$

find the smallest exponents N in each case.

F. Determine the sum of all the integers greater than 1000 which may be formed using only the digits 1, 3, 5, 7, no digit being repeated in any given integer.

G. Let $P(N) = 1 \cdot 2 \cdot 3 \cdot \cdots \cdot N$
$S(N) = 1+2+3+\ldots+N$

For exactly what values of N does S(N) divide P(N) with remainder zero?

H. If A is a positive integer, then there is an integer N such that the high order digits of factorial N are the integer A. Find the smallest N for each A between 1 and 1000.

I. According to one of the standard tests for randomness, the numbers π and e each contain approximately the same number of digits 0, 1, 2, 3,...,9 when sufficient digits of each number are taken.

$$\pi = 31415926535897932384626433832795028841971 6$$
$$e = 27182818284590452353602874713526624977572 4$$

So, if we go far enough out in e, we should be able to rearrange the first K digits to provide M place accuracy of π.

To get $\pi_1 = 3$, we need 18 digits of e, but by the time we get 18 digits of e we actually have

$$\pi_7 = 3141592$$

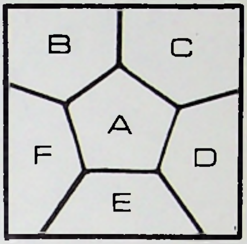
and 21 digits of e gives

$$\pi_{12} = 314159265358$$

or does it?

Find a value of K such that the first K digits of e can be rearranged to form the first K digits of π, or find as small a D as possible such that K+D digits of e contain K digits of π.

J. In the accompanying figure, A is a regular pentagon whose center coincides with the center of the unit square, and one side of the pentagon is parallel to a side of the square. The five radiating lines pass through the center of the pentagon. What are the coordinates of the five vertices of the pentagon, to make the six areas equal? Solve the same problem for regular 6, 7, 8,... 64 sided polygons in the unit square.



K. A common problem of logical ability on intelligence tests is to furnish a number of terms of a sequence, for which the testee is to add the next term or terms. For example, given the sequence

5, 11, 17, 23, 29,

it would be reasonable to respond 35 and 41, since the given data is in progression with a common difference of 6.

For the sequence

5, 11, 17, 23, 29, 30,

there seems to be no simple pattern. One could, of course, fit a 5th degree polynomial to the given data:

$$y = (-1/24)x^5 + (5/8)x^4 - (85/24)x^3 + (75/8)x^2 - (65/12)x + 4,$$

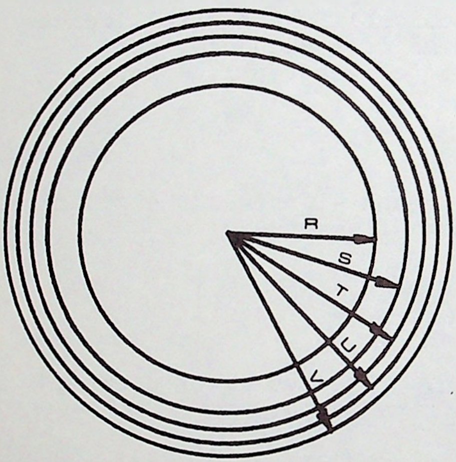
and obtain 11 and -58 as the next two terms. A simpler explanation is that the given values are the number of zeros that cannot appear at the low order end of the factorials. That is, there is no factorial ending in 5 zeros, or 11 zeros, or 17, 23, 29, 30,... zeros.

Problem: extend the sequence for the next 20 terms.

L. In the Journal of Recreational Mathematics, Vol. 6, No. 1, page 72, there appeared Dave Silverman's problem of the Fortran Flip Function. In the list to the right, the value of f(N) is the number of times that N appears in the f column. Mr. Silverman asked for a Fortran program to print this function line-by-line; that is, to rule out building up the function in storage by the same scheme one might use by hand.

Our problem is to draw a flowchart for the logic of developing the Flip Function on a line-by-line basis.

N	f(N)
1	[illegible]
2	2
3	2
4	3
5	3
6	4
7	4
8	4
9	5
10	5
11	5
12	6
13	6
14	6
15	6
16	7
17	7
18	7
19	7
20	8
21	8
22	8
23	8
24	9
25	[illegible]
26	[illegible]
27	9
28	9



M. In the figure at the left, the inner circle has an area of 1. The area of the ring between the inner circle and the circle of radius S is 1/2. The next ring has an area of 1/3, and so on: the Nth annulus has an area of 1/N. The successive radii converge--to what value?

N. The following equations follow the rules laid out in freshman algebra texts:

$$16 - 36 = 25 - 45$$

$$4^2 - 4\cdot 9 = 5^2 - 5\cdot 9$$

$$4^2 - 4\cdot 9 + (9/2)^2 = 5^2 - 5\cdot 9 + (9/2)^2$$

$$(4 - 9/2)^2 = (5 - 9/2)^2$$

$$4 - 9/2 = 5 - 9/2$$

$$4 = 5$$

Has any rule been violated? Has an extraneous root been introduced?--where? Just what is the flaw?

O. Divide the number 10 into two parts,

$$10 = X + Y$$

such that the product of each part plus its square root is 23; that is,

$$(X + \sqrt{X})(Y + \sqrt{Y}) = 23.$$

The value of X is .93701...; find X correct to more decimal places. (Due to Kraitchik.)

P. Find a solution in integers to the equation

$$y^3 - 117x^3 = 5$$

or show that no such solution exists.

Layman's Guide to Computing

JOHN G. SCOTT

People working in the field of computing are regarded as strange by the general public. Strangers insist on pressing for an answer to the question "What do you really do?" Those in hardware have it easier at cocktail parties, for example, since they can avoid the situation by saying they are "in electronics," but, by and large, a professional computist or computerist is at a disadvantage socially.

A number of obvious reasons come immediately to mind. First, many people know very little about computing. Many find simple math puzzling; data processing is then beyond comprehension.

Next, the distorted role of computers in modern entertainment--books, television, and film--has misinformed an already bewildered public. In print or on the screen, from Rossum's Universal Robots (Capek, 1920) to the modern cyborg/computer craze; from Fritz Lang's Metropolis (1926) through Desk Set (1957) to the Kubrick/Clarke 2001: A Space Odyssey (1968); many of the same strange notions appear. One hears or reads the phrase "the computer" spoken as though no more needed to be said. Not any computer, necessarily, *the* computer. How many times have you read or heard this line of dialogue: "We must flee, the computer has taken over."? No one ever questions this statement. It is generally assumed that an inanimate machine can "take over." Even granting this for the sake of a good plot, is there not one person who might think of pulling the plug? Of course, in some cases, the gimmick was that the plug had been sealed, so anyone who tried to pull it got clobbered, or something.

Television newscasters and the press in general have not helped, with their constant references to what are commonly called computer errors. We are regaled with phrases such as "the computer goofed" with no mention of the human being(s) who fed information into the machine in the first place. You never read "police are looking for a Chevrolet..." but always "the *driver* of a Chevrolet." The press has not learned to transfer this notion over to "a massive foul-up in airline reservations in Ireland yesterday was attributed to *programming* error..."

Television performs another insidious service in the broadcasting of deceitful commercials. The for-profit computing schools are falling all over themselves in an effort to sucker as many high school dropouts as possible into shelling out X number of dollars to be able to enter the "high-paying world of computer programming." The worst at present is the one with the line that goes

"Maybe you're like me--the kind of guy who likes workin' with his hands, but who doesn't want ta get his finger-nails dirty."

Matters are sometimes made worse when one attempts to uncover some facts about the industry. The incredible diversity of human endeavor in which computers play a role is overwhelming. It is amazing enough that computers are commonplace in fields ranging from aviation to zoology, but downright mind boggling for the layman to learn of their use in designing jeans and routing the delivery of the morning paper.

It appears that many people know more about astrology than computing. And like it that way. This is somewhat strange, since it takes a modicum of effort to learn the jargon and manipulations of astrological data, and some pretty dumb people manage to do that. Computers are viewed with disdain by some, fear and suspicion by others. Much as those who deal with stocks come to regard compilations of buying and selling prices in personified form--"the market is reacting to the President's latest cabinet reshuffling," or "the market is sluggish today"--many refer to computers as though the term described a species apart.

Perhaps a book is needed, a good book, introducing the layman to computers and computing. Not only could an author and publisher help raise the average knowledge concerning this business, but with a cover displaying a scantily-clad woman lounging next to a 370/158 under the title "Layman's Guide -- to Computing," a lot of money could be made. If this kind of advertising works for poor products, perhaps it would be successful for a good one.

Of course, such a book is useless if no one reads and understands it, and the one main distressing fact is that most people find it easier to cope with their ignorance. This is a dangerous condition. At the very least, it leads to exploitive TV ads. Continued misunderstanding of the potentialities of computing can lead to many other ills, including a surrendering of privacy.

Log 16	1.20411998265592478085495557889797210707275952584843
Ln 16	2.77258872223978123766892848583270627230200053744102
$\sqrt[3]{16}$	2.51984209978974632953442121455645670114050292940301 59601639502243105993530279189674587931248725101883 08620512071231330518798048081227474456918220608538 71049392128523325000195494905313096061373437081103 73784917450335283987474193901967655663227983102586 27390732367894926897153140606238191791969482211962 32581410718163295602294704265096954259576048441716 41065159450533244053380113312163989431256352810121 32965354714534083897241524288593138841015863834488 29618408964656802549406439285641624038114283779929 19996635007603777379188404111844042309459947697605 21472739483577558431596935019907926015652191924840 69664773202797147268678194742530559919839399367558 26336336308857700559303058556215359428004081211349 60787712250343671401381596999268395258294808966908 05394309524570263560412877560952986451581057969341 71610572516260010858777121441219494446081262714469 87291681315183383383345412024880579340000213816207 06277058054008301684646724797787729935643882996760 54145914353625758002891492454295404696714303811013
$\sqrt[5]{16}$	1.74110112659224827827254003495949219795825084869601
$\sqrt[7]{16}$	1.48599428913694842479985328671459260632371135943711
$\sqrt[10]{16}$	1.31950791077289425937400197122964013303346901319342
$\sqrt[100]{16}$	1.02811382665606650934634495879263497654868284295379
e^{16}	8886110.52050787263676302374078145035080271982185663 88397839883170489809373214781504432221066858
π^{16}	90032220.8429332795671307682279165370944602964463416 2447628776528827142603194849927545204026320
$\tan^{-1} 16$	1.50837751679893927075734257865424632849231081189005 37158799444792552179359263010415000480509476524179
16^{100}	2582249878086908589655919172003011874329705792829223 5128306593565406476220168411946296453532801378314359 03171972747493376

16